DAYANANDA SAGAR COLLEGE OF ENGINEERING
(An Autonomous Institute affiliated to Visvesvaraya Technological University (VTU), Belagavi,
Approved by AICTE and UGC, Accredited by NAAC with 'A' grade & ISO 9001 – 2015 Certified Institution)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560 111, India

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Skill Development Program on Signal & Image processing with Embedded hardware integration

From:25/03/25 To:27/03/25

Report on: Implementing a Walking Step Counter Using Accelerometer Sensor

Submitted by:

S.No	Name	USN
1.	Sanjana CH	1DS23EC189
2.	Nivedita N Devakari	1DS23EC143
3.	Sairam	1DS23EC184
4.	Poornachandra PU	1DS23EC152
5.	Advait RP	1DS24EC401
6.	Harish R	1DS24EC406

Table of Content:

S.No.	Topic	Page No.
1.	Abstract	2
2.	Introduction	2
3.	Problem statement & Objectives	3
4.	Methodology	4
5.	Implementation in MATLAB	5
6.	Results and Discussion	6-7
7.	Conclusion	8
8.	Future enhancements may include	8
9.	References	8

1.Abstract:

The increasing demand for smart health monitoring and fitness tracking systems has led to the development of efficient step counting mechanisms using motion sensors. This project focuses on the implementation of a walking step counter using accelerometer sensor data in MATLAB. The objective is to process and analyse raw acceleration signals to accurately detect and count steps taken during walking. In this approach, raw tri-axial acceleration data (X, Y, Z) is collected from an accelerometer sensor and imported into MATLAB for analysis. The magnitude of the acceleration is computed to simplify detection and make the system independent of the sensor's orientation. To improve signal quality and reduce the effects of noise and gravity, a filtering process—consisting of high-pass and low-pass filters—is applied. The filtered acceleration signal is then analysed using a threshold-based peak detection algorithm, where each peak above a defined threshold, with appropriate time separation, is counted as a valid step. The implementation in MATLAB allows for efficient data visualization, parameter tuning, and validation of the algorithm against real-world walking data. The results confirm that the MATLAB-based step counting system is effective and provides an accurate estimation of steps during normal walking activity. This work forms the basis for future enhancements such as adaptive thresholding, real-time processing, and integration with mobile or wearable health monitoring systems.

1. Introduction:

Step counting has become an essential feature in fitness trackers, smartphones, and wearable health-monitoring devices, offering users a simple yet powerful way to monitor their physical activity. Accurate step detection not only promotes healthier lifestyles but also serves as a valuable metric in medical rehabilitation, elderly care, and sports performance analysis. Among the various sensors available, accelerometers are widely used due to their low cost, small size, and ability to measure acceleration across three axes (X, Y, and Z), capturing even subtle body movements.

In this project, we implement a walking step counter using accelerometer sensor data in MATLAB. The raw acceleration signals are processed to compute the magnitude of the acceleration vector, which provides a direction-independent representation of motion. This magnitude signal is then filtered to remove high-frequency noise and the constant influence of gravity. By applying a peak detection algorithm with thresholding and timing constraints, the system is able to count steps based on the periodic patterns of walking.

2.1.Problem Statement:

- Traditional step counters may give inaccurate results.
- Accelerometer data is often noisy and hard to interpret directly.
- Gravity and random movements affect step detection accuracy.
- Orientation of the sensor can impact the readings.
- A reliable algorithm is needed to detect steps from raw data.
- The system must work across different walking speeds.
- Real-time processing and accuracy are important.
- MATLAB is to be used for processing and visualization of results.

2.2 Applications of Walking Step Counter Using Accelerometer Sensor

- Fitness tracking in smartwatches and wearable devices
- Monitoring daily physical activity and step goals
- Elderly movement monitoring and fall detection systems
- Patient progress tracking in rehabilitation and physiotherapy
- Integration in mobile health and fitness applications
- Gait analysis for detecting walking abnormalities or disorders
- Activity recognition systems (walking, running, sitting, etc.)

2. Objectives:

- To develop a step counting system using accelerometer sensor data.
- To process raw acceleration signals and compute their magnitude
- To filter out noise and the effect of gravity from the acceleration data
- To implement a peak detection algorithm for accurate step identification
- To ensure reliable step counting across different walking speeds
- To build the entire step detection system using MATLAB environment

3. Methodology:

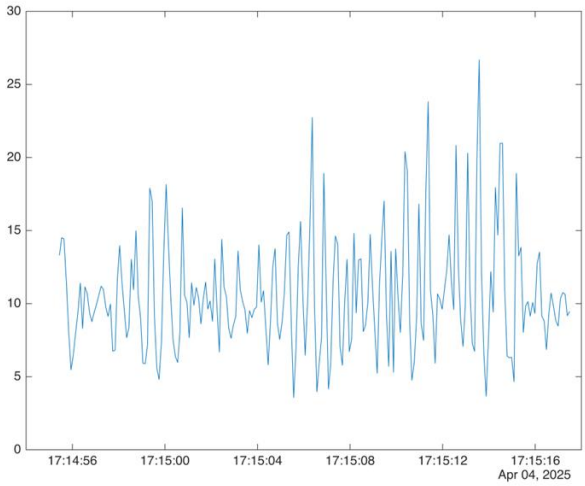
- **Data Collection**: Acquire tri-axial accelerometer data (X, Y, Z) using a sensor or dataset. Ensure the data includes normal walking motion.
- **Preprocessing:** Convert data to consistent units and format. Resample if necessary to maintain a fixed sampling rate.
- **Compute Acceleration Magnitude:**
- **Filtering:** Apply a high-pass filter to remove gravity effects. Optionally apply a low-pass filter to reduce high-frequency noise.
- **Peak Detection:** Implement a threshold-based peak detection algorithm. Set a threshold value to detect significant peaks related to steps. Add a time constraint (e.g., 300–700 ms between steps) to reduce false detections.
- **Step Counting:** Count valid peaks that satisfy both threshold and time conditions.

  Store or display the total step count.
- **Visualization:** Plot raw and filtered signals in MATLAB. Highlight detected steps for verification.
- **Evaluation:** Compare detected steps with actual steps taken. Adjust thresholds and filters to improve accuracy if needed.

Implementation in MATLAB and Results :

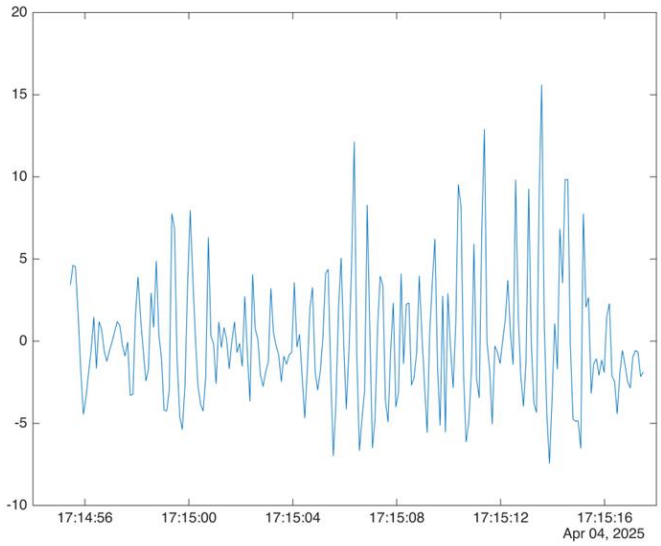
Step-1: Load Step signal and plot signal data

```
load hdjdkek.mat
% Extract X, Y, and Z columns
ax = Acceleration.X;
ay = Acceleration.Y;
az = Acceleration.Z;

% Compute Acceleration Magnitude
acc_magnitude = sqrt(ax.^2 + ay.^2 + az.^2);
fs=10;
% Add the computed magnitude to the timetable
Acceleration.Magnitude = acc_magnitude;

plot(Acceleration.Timestamp,Acceleration.Magnitude)
```



Step-2:Eliminate the trend in signal data

```
designal=detrend(acc_magnitude);
plot(Acceleration.Timestamp,designal);
```

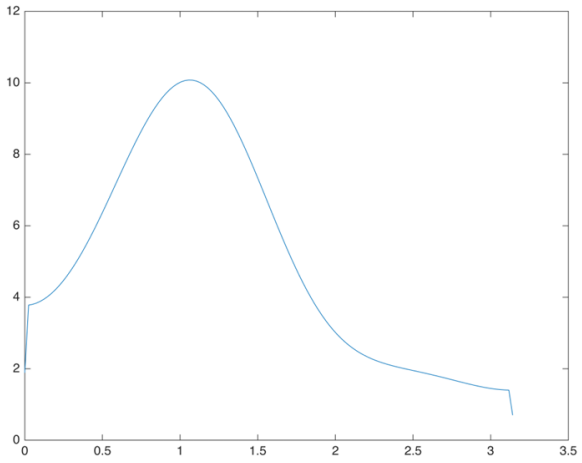


Step-3:Convert time domain to frequency domain

```
[pxx,fxx]=pwelch(designal,fs)
```

```
pxx = 129×1
    1.8871
    3.7810
    3.8016
    3.8359
    3.8838
    3.9452
    4.0200
    4.1081
    4.2092
    4.3232
     ⋮
fxx = 129×1
         0
    0.0245
    0.0491
    0.0736
    0.0982
    0.1227
    0.1473
    0.1718
    0.1963
    0.2209
     ⋮
```

```
plot(fxx,pxx)
```



Step 4-: Design low pass filter with automatic coding

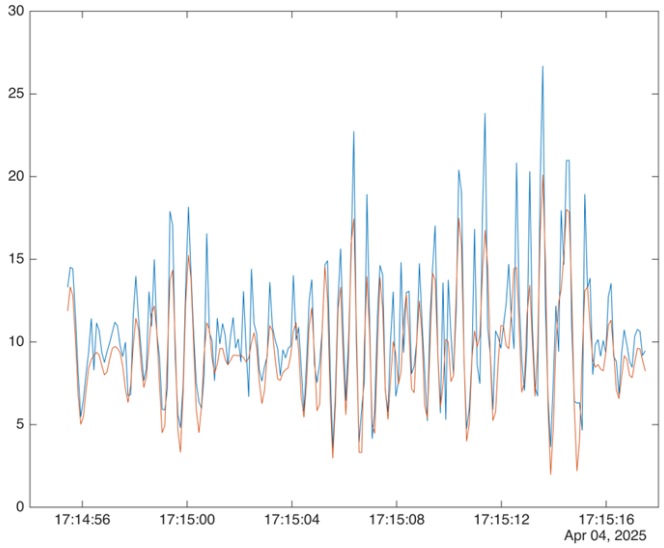
```
fs = 10;        % Sampling frequency (adjust if needed)
fc = 2;         % Cutoff frequency (Hz)
N = 4;          % Filter order
Rp = 0.5;       % Passband ripple (dB)

% Design Chebyshev Type I Low-Pass Filter
[b, a] = cheby1(N, Rp, fc/(fs/2), 'low');  % Normalize cutoff frequency
```

Apply Filter to the signal

```
filtered_data = filtfilt(b, a, acc_magnitude);  % Zero-phase filtering
figure;
```

Time Domain Visualisation

```
plot(Acceleration.Timestamp,Acceleration.Magnitude,Acceleration.Timestamp,fi
ltered_data)
```



Step 5:Peak finder

```
[peaks,locs]=findpeaks(filtered_data)
```

```
peaks = 38×1
   13.3283
    9.3762
    9.7344
   11.4321
   12.1804
   14.3390
   15.2497
   11.1689
    9.6160
    9.2059
      ⋮
locs = 38×1
     2
    12
    19
    27
    34
    41
    47
    54
    60
    65
      ⋮
```

Step 6:Total steps

```
xy=length(locs);
disp(xy)
```

```
    38
```

Conclusion:

This project demonstrates a simple and effective method to count walking steps using accelerometer data in MATLAB. By calculating acceleration magnitude, applying filters, and detecting peaks, the system accurately identifies steps during walking. The use of MATLAB enables easy data processing and visualization, making it suitable for analyzing motion patterns. This approach can be further enhanced and integrated into wearable health and fitness applications.

7. Future enhancements may include:

- Implement real-time step counting using live sensor data
- Use adaptive or dynamic thresholding for better accuracy
- Add detection for other activities like running or climbing stairs
- Estimate step length and total distance walked
- Integrate with GPS or mobile health apps
- Include fall detection for elderly care
- Develop a user-friendly GUI in MATLAB
- Store data in the cloud for long-term analysis
- Improve performance using machine learning algorithms

8. References:

- MathWorks. (2023). Signal Processing Toolbox Documentation. Retrieved from https://www.mathworks.com/help/signal
- MathWorks. (2023). Importing and Analyzing Sensor Data. Retrieved from https://www.mathworks.com/help/matlab/importing-data.html
- Ravi, D., Wong, C., Lo, B., & Yang, G.-Z. (2017). A deep learning approach to on-node sensor data analytics for mobile or wearable devices. IEEE Journal of Biomedical and Health Informatics, 21(1), 56–64.
- Mannini, A., & Sabatini, A. M. (2010). Machine learning methods for classifying human physical activity from on-body accelerometers. Sensors, 10(2), 1154–1175.